面向数据中心异构负载的低成本协同资源供应方法和系统[1]

詹剑锋 王磊 李晓娜 施巍松 翁楚良 张文耀 臧秀涛

翻译 周俊平

摘要：最新的研究分析表明，服务器购置费用仍然是影响大规模数据中心或云系统开销的主要因素。在本文中，我们认为经典的资源供应问题已经发生显著变化：异构负载已经成为大规模数据中心的常态，而目前的资源供应方法并不能很好地应对这种异构性。本文报道了我们针对这一问题展开的工作。我们的贡献有以下三个：首先，我们提出了一个协同资源供应的解决方案，通过充分利用异构负载的差异，降低在竞争条件下的尖峰负载的资源消耗；第二，基于四种典型的异构负载：并行批处理作业、Web 服务、搜索引擎和 MapReduce 作业，我们构建了支持异构负载的 PhoenixCloud 系统，能够有效地进行协同资源供应；第三，我们通过使用真实和合成的负载日志进行实验对系统进行了综合的评价。实验结果显示，我们的解决方案与现有的非协同解决方案相比可以极大节省服务器购置成本，该方案可以广泛用于当前的托管数据中心或云系统中，如利用统计复用技术的 EC2 系统或实现了初步弹性资源供应技术的 RightScale 系统。

关键字：数据中心，云，协同资源供应，统计复用，成本，异构负载

1 引言

当前越来越多的计算和存储模式正在从类似 PC 的客户端向数据中心[32]或（公共）云[21]系统转移，典型的例子有 EC2 服务和 Google Apps。转向服务端计算的驱动力不仅是用户的需求，如管理方便（无需配置或备份）和使用方便（支持通过浏览器[32]的访问），还有大规模数据带来的规模经济效益：大规模规模数据中心[32]的成本相对于小规模数据中心可以节约 80%~90%[20][29]。然而，大规模数据中心成本仍然很高。据报道在亚马逊[29] ，15 兆瓦（MW）的托管数据中心电力成本高达每月五百六十万美元。因此如何降低数据中心的成本对于资源提供商是非常重要和紧迫的问题。

对于亚马逊和谷歌等数据中心的成本模型研究[32][44][46][51]获得了两点结论：第一，不同于传统的企业系统，数据中心的人力成本因素已经从原来在总成本占很高比例转变到现在几乎无关紧要[29]。第二，服务成本中的服务器硬件成本贡献了最大的份额（超过一半），其他成本还包括电源及冷却基础设施的成本、电力成本及其他基础设施的成本，这些成本很大程度上也和服务器数量有关。针对这种情况，本文的重点是如何降低对服务器的需求，这是影响数据中心成本中最重要的因素。

目前的降低服务器需求的解决方案可以分为两类：第一，**基于虚拟化聚集**的方式，其目的是：（1）. 解决应用程序隔离或异构操作系统所造成的服务器扩张问题（即由于未充分利用应用的服务器聚集特性，造成数据中心占用的空间和消耗的服务器或存储资源快速增长）；

[1]原文发表于IEEE Transaction on Computers，标题为*Cost-aware Cooperative Resource Provisioning for Heterogeneous Workloads in Data Centers*

（2）. 在托管数据中心，为多层服务[48] [49]或科学计算负载[20]提供弹性资源。第二，**在 EC2 的系统中，利用统计复用技术**[27]，以减少对服务器的需求。复用技术意味着多个用户之间共享资源 [58]，与根据所有用户的峰值需求来分配资源的方法不同[58]，统计复用技术允许所有用户峰值资源需求的总和超过数据中心总容量（就像在机票预订系统中，经常超售资源[39]，以最大限度地提高销量），从而将多个轻负载的应用以更高的利用率聚集在一个给定的硬件系统上。

由于越来越多的计算移动到数据中心，资源提供商必须面对不同类别异构负载（例如 Web 服务器、数据分析作业、并行批处理作业）增加引起的服务器扩张的挑战。这种现象已经受到越来越多的关注。例如，人们注意到在 Nutch（http://nutch.apache.org/）搜索引擎系统中包括两个主要异构负载：类 MapReduce 并行数据分析和搜索引擎服务；在 2010 年 10 月 30 日北京举行的 IDC HPC 用户的论坛上，波音公司也报道了这一趋势，他们将并行批处理作业和 Web 服务应用程序合并到一个数据中心上。异构负载往往有着显著不同的资源消耗特点和性能目标，我们将在第 3 章讨论这一问题。由于异构负载的增长，服务器扩张所造成的挑战，已不能简单地由以前的基于虚拟化的聚合扩展来解决。

而且，只利用统计复用或超售资源技术也不能有效地应对异构负载增加造成的服务器扩张的挑战。受制于电源和其他因素，数据中心的规模不能无限扩大。在大量用户和服务可以随时部署或取消部署的环境下，类似 EC2 一样利用统计复用技术使得系统资源利用率曲线保持平滑是不可能。此外，超售机制对于资源供应是一个概率模型，因此它不适合关键服务。

从已有的成本分析数据[32][44][46][51]中可知，服务器成本约占数据中心总成本的 53％[29]，而服务器成本是由系统的容量决定的，并且电力与基础设施成本也和系统容量密切相关。由于系统容量必须至少比负载的资源消耗峰值大，所以资源消耗峰值是系统容量的下界，因此，我们可以通过减少资源需求的尖峰负载，以节省服务器成本和电力以及基础设施的成本，从而降低整个数据中心的成本。

我们根据资源消耗的特点和性能目标，充分利用异构负载的差异性，提出了协同的资源供应解决方案，以降低数据中心负载的资源消耗峰值。我们的贡献包括三个方面：

首先，我们利用异构负载的差异，提出了一个协同的资源供应解决方案，支持服务提供商在竞争条件下运行异构的负载，并在更高的粒度——两个异构负载的组合，利用统计复用技术，以节省服务器成本。

第二，我们基于以前的工作[20] [21]实现了一个创新系统——PhoenixCloud，为四种有代表性的异构负载：并行批处理作业、Web 服务器、搜索引擎、MapReduce 作业提供协同供应资源。

第三，我们对 PhoenixCloud 系统和类 EC2+RightScalelike 系统进行了综合评价比较。

本文的其余部分安排如下：第二章主要描述问题的重要性；第三章主要提出了协同资源供应的解决方案；第四章详细描述了我们的协同资源供应解决方案；第五章对系统进行了全面地评估和比较；第六章列举了相关工作；第七章总结。

2 问题定义和动机

根据文章[29] [32] [46]的观点，服务器的购置成本占总数据中心总成本很大部分。从成本分析模型 [29][32] [46]来看，系统的容量大小是降低数据中心成本的关键因素：一方面，最大的一

部分成本，即服务器成本（53％）[29]是由服务器的系统容量决定的；另一方面，第二大部分的数据中心成本，即电力和冷却基础设施的成本取决于最大设计（额定）功耗，这个值是由系统容量、数据中心的容错度和功能需求所共同决定的。

由于系统容量必须至少大于负载资源消耗的峰值，因此，我们必须减少尖峰负载的资源需求，从而降低服务器的成本和电源及冷却基础设施的成本。由于每个事件请求释放或供应资源将触发所设置的动作，例如删除操作系统或数据，在类 EC2 系统中，弹性资源供应将导致管理上的开销，管理开销可以由下面的公式来衡量：

管理开销=（累计调整资源数量）×（每次调整的平均消耗时间）

资源提供商的资源消耗总量是有效资源消耗之和，而有效的资源消耗是负载的直接消耗与管理开销之和。

我们对尖峰资源消耗和有效的资源消耗的准确定义如下：

设服务提供商的负载是 $\boldsymbol{w}_{i,j=1\cdots N}$，对于 $\boldsymbol{w}_i$，整体运行持续时间是 $\boldsymbol{T}_i$，在第 $\boldsymbol{j}$ 时间单位 $(\Delta\boldsymbol{t})$，$\boldsymbol{j}=1,\cdots,\boldsymbol{M}$，资源调配给 $\boldsymbol{w}_i$ 的服务器数量是 $\boldsymbol{C}_{i,j}$，则**尖峰资源消耗**为 $(\sum_{i=1}^{N}\boldsymbol{C}_{i,j}, \boldsymbol{j}=1,\cdots,\boldsymbol{M})$ 的最大值，而**有效的资源消耗**是 $\sum_{i=1}^{N}\sum_{j=1}^{T_i/\Delta t}\boldsymbol{C}_{i,j}\Delta\boldsymbol{t}$ 。

当用户租赁资源时，费用按照单位时间的粒度—— $\Delta\boldsymbol{t}$ 收取。例如，在 EC2 上，租赁资源的单位时间为一小时。因此，我们可以使用负载的有效资源消耗，即租赁资源的大小乘以其租赁时间来间接地反映服务提供商的成本，作为标杆指导降低成本的工作。降低服务器购置成本的同时不能牺牲服务提供商和最终用户的服务质量，而反映质量的性能指标和不同的负载密切相关：对于并行批处理作业或 MapReduce 作业，主要关注点是服务提供商在完成工作时的吞吐量[3][8]；而最终用户关心的是每个作业的平均周转时间，即从作业提交到完成[8][10]的平均时间。对于 Web 服务器和搜索引擎，服务提供商主要关注的是以每秒响应的请求数来衡量的吞吐量[5][7]，而最终用户主要关注每个请求的平均响应时间，以此作为服务质量的主要指标[5][7]。

在此背景下，本文着重讨论在不显著影响服务提供商和最终用户性能指标的情况下如何降低服务器成本。如上所述，以节点数量表示的最小系统容量必须要至少大于负载的尖峰资源消耗，所以我们用负载尖峰资源需求来衡量服务器最低成本。我们的资源供应解决方案专注于如何降低尖峰资源需求，同时又不会严重降低服务提供商和最终用户的性能指标。

为什么以前的工作未能提出一个协同解决方案？

首先是因为用户通常为并行批处理作业或 Web 服务[20]选择专用的系统，并为它们对应的交互式/网络负载和批处理负载安排不同的调度机制；此外，在 http://www.cs.huji.ac.il/labs/parallel/workload/或 http://ita.ee.lbl.gov/html/ traces.html，可以获取公开可用的高性能计算（HPC）或 Web 服务的负载数据，因此大多研究者都选择把如何优化同构负载作为研究目标。

其次，云计算是近年来兴起的新平台。在亚马逊的 EC2 成为公用设施之后，人们才开始认识到越来越多的异构负载会出现在同一个平台上。

第三，在同一平台上聚集异构负载的日志数据是不公开的，因此，研究人员没有公开可用的基准测试程序和负载数据(trace)来评价他们的解决方案，这严重阻碍了他们深入研究异构负载的协同资源供应问题。为克服这一障碍，我们发布了云计算[59]的基准测试程序集合。

EC2 使用统计复用技术，充分地利用了异构负载差异性。我们的工作进一步深入，在更高的粒度——两个异构负载的组合层次上，充分利用统计复用特性，以节省服务器成本。

3 协同资源供应模型

随着越来越多的计算移动到数据中心，资源提供商对于越来越多的异构负载需要有相应的资源供应手段。不同于应用隔离或操作系统异构性[50]（服务器聚集）所造成的服务器扩张，不断增加的负载类型和不断增长的负载强度对系统容量规划提出了新的挑战，因为它们的资源消耗有显著不同的特点。

在本文中，针对批处理作业、Web 服务器、搜索引擎、MapReduce 作业， 我们利用负载的差异性，在两个方面节省服务器的成本。

首先，这四种负载在时间、强度、规模、持续时间和资源使用模式方面的要求是显著不同的。Web 服务器的负载通常由一系列持续时间短（如 1 秒之内的）的请求组成，尖峰负载与平均负载的比值很高，通过资源复用，可以为多个请求同时提供服务和交错提供服务；批处理作业的负载由一系列资源大小需求有所不同的作业组成，每一个作业是并行或串行的应用程序，串行应用的运行时间更长，以小时计算，运行并行应用程序需要一组独占的资源；虽然 MapReduce 作业同用 MPI[2]编写的批处理作业有很多相似之处，但是它们有两个显著的差异：（一）MapReduce 作业往往是由各种不同大小的数据输入组成，数据输入决定其资源的需求规模；（二）在运行时，MapReduce 的任务是相互独立的，所以杀掉一个任务不会影响另一个[28]。这种任务之间的独立性和 MPI 批处理作业的编程模型（MPI 任务并发执行并在执行过程中存在消息交换）形成鲜明的对比。

其次，这四种负载的性能指标是不同的。利用这一点，我们可以协调运行异构负载的服务提供商的资源供应行为，以减少峰值资源消耗。在一般情况下，对于批处理的最终用户而言，提交作业时若无资源可用可以排队等待。然而，Web 服务，如 Web 服务器或搜索引擎，每一个独立请求需要立即响应。因此我们可以首先满足 Web 服务资源请求，而拖延执行并行批处理作业或 MapReduce 作业。

对于协同资源供应，我们提出如下两项指导原则：

（1）为安全或隐私的考虑，如果服务提供商不允许协同的资源供应，或无法找到一个协同的服务提供商，资源提供商将为其负载独立供应资源；（2）如果服务提供商允许，则资源提供商以包括两个异构负载的组合为粒度支持协同的资源供应。

对于每组异构负载，我们提出了以下协同的资源供应模型:

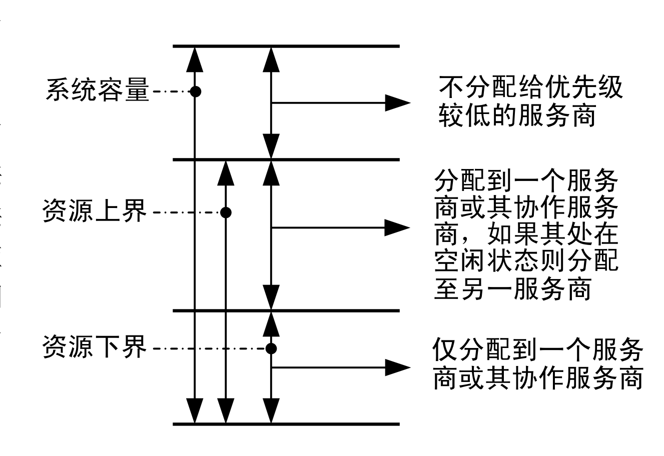


图1. 系统容量和两个资源边界

首先，当服务提供商依赖于托管数据中心，它需要指定两个资源范围：资源上界和下界。资源提供商保证在资源下界中的资源只能被分配到指定的服务提供商或其协同服务提供商。

[2] Message Passing Interface，一种基于消息传递的并行程序设计标准

介于资源上界和下界的之间资源，资源提供商首先满足指定的服务提供商或其协同的服务提供商的资源请求，但是两个边界之间的资源处于闲置状态时可以被重新分配到另一个服务提供商。图 1 显示了系统的容量和两个资源边界之间的关系。

第二，资源提供商是否允许一个由两个服务提供商组成的候选组合加入服务对象决策过程如下：

资源提供商设置和维护的资源预订阈值（RBT[3]）作为决定是否允许两个服务提供商候选人加入组合的考量因素之一。当资源预订阈值大于 1，意味着资源被超售。通过设置合理的资源预订阈值，资源提供商保证所有加入系统的服务提供商和两个即将加入的服务提供商（候选人）的资源需求上限总和不超过系统容量乘以资源预订阈值。除此以外，两个服务提供商候选人资源下界的总和必须低于目前系统的闲置资源的大小。只有当两个条件同时满足，这两个服务提供商候选人才能加入组合。

第三，如果一个组合被允许加入，组合内的两个服务商在运行异构的负载时和启动时资源提供商都会为其分配数量在资源下界以内的资源。

第四，对于每组两个异构负载：资源请求需要立即做出反应的那个负载比资源请求可以排队的负载具有更高的优先级。例如，对于两个典型的异质性负载：Web 服务器和并行批处理作业，Web 服务器比批处理作业具有更高的优先级，因为后者在没有资源可用时提交的作业可以排队，而 Web 服务对每个请求都需要立即做出反应。

我们提出如下运行时算法（详见附录 A 算法 1）：

0. 资源提供商设置一个使用率参考值（URR）和一个比例共享因子（PSF），作为在竞争环境下协调资源共享的依据。我们定义比例共享因子（PSF）为请求资源的低优先级服务提供商的资源下界（LB）与所有加入的较低的优先级的服务提供商的资源下界的总和的比值。使用率为所有服务提供商分配的资源与系统的容量的比值

1. 从请求队列中提取一个请求

2. 如果一个具有较高优先级的服务提供商请求资源，资源提供商将按照请求的大小立即分配给它。然而，如果一个具有较低优先级的服务提供商请求资源，则需要进一步分析如下：

   i. 对于一个较低优先级的服务提供商，如果其所被分配资源的大小加上其请求资源超过了其应有的资源上限，将没有资源分配给它；

   ii. 否则服务提供商将被授予一部分所要求的资源，具体如何分配由资源提供商通过用使用率参考值与使用率比较来决定。计算方法如下式：

   AllocatedSize = MIN{所要求资源的大小，未分配资源的大小}×PSF。

   其中，AllocatedSize：分配给服务商的资源量；PSF：比例共享因子

3. 当使用率不比使用率参考值大的时候，如果所请求的资源的大小小于未分配资源的大小，资源提供商授与服务提供商所要求的大小的资源，否则资源提供商将给予服务提供商资源的大小按下式计算：

   AllocatedSize =（未分配资源的大小）×PSF。

[3] resource booking threshold

4. 查看请求队列是否还有未处理的请求。如有则转回步骤 1，否则结束。

第五，服务提供商根据自己的资源要求主动决定请求或释放资源。

最后，如果服务提供商不允许协同资源供应或不能找到一个协同的服务提供商，我们可以将协同的服务提供商看作空值并作为一个独立的资源供应解决方案简化上述模型。

当我们假定的资源提供商有无限的资源，资源预定阈值和使用率参考值可以分别设置为 1。这种情况下，表明对于服务提供商的资源请求该提供商的资源是足够的，所以资源售是没有必要的并且每个请求资源远小于系统容量。

4 PhoenixCloud 的设计和实现

在本节中，在我们以前的系统 DawningCloud [20] [21]基础上，我们给出了 PhoenixCloud 系统的设计和实现。目前，PhoenixCloud 支持 Web 服务器、并行批处理作业、搜索引擎、MapReduce 作业之间的协同资源供应。它可以扩展使用于其他数据密集型的编程模型，例如，类 Dryad 数据流和 All-Pairs（这些都是我们以前实现的 Transformer 系统所支持的[33]）。

PhoenixCloud 存在如下两个层次架构：一个为资源提供商的公共服务框架（CSF，Common Service Frame），另一个为服务提供商的瘦运行环境软件（TRE，Thin Runtime Environment），它为每个服务提供商管理资源和负载。两个层次架构意谓着公共服务框架和瘦运行环境是分离的：公共服务框架由资源提供商供应和管理，独立于任何瘦运行环境。在公共服务框架的支持下，可以按照需求创建一个瘦运行环境或两个协调的瘦运行环境。瘦运行环境概念意谓着对于异构负载，其对应的运行时环境具体功能集合由公共服务框架实现，而瘦运行环境只为负载实现共同的核心功能。

图 2 描述了 PhoenixCloud 系统的体系结构视图，其中一个并行批处理作业或 MapReduce 作业的瘦运行环境（PBJ[4] TRE）和一个 Web 服务器的瘦运行环境（WS TRE）复用了公共服务框架。在此，我们只是简单地给出了公共服务框架和瘦运行环境的主要组成部分，其他组件的细节可以在我们公开提供的技术报告[35]中找到。

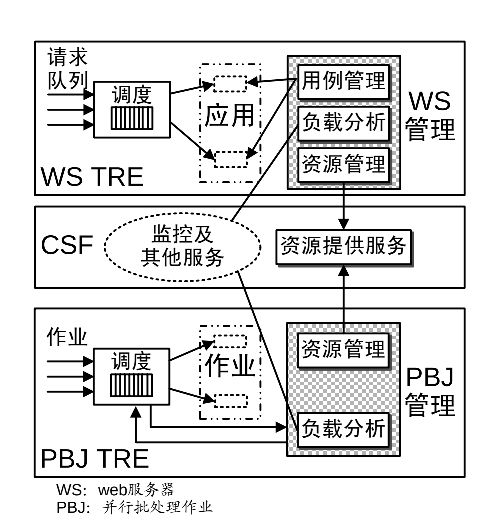


图2. 并行批处理作业瘦运行环境及web服务瘦运行环境与公共服务框架的交互

在公共服务框架中，资源供应服务负责协调资源供应。框架有两种类型的监视器：资源监视器和应用监视器。在每个节点上的资源监视器监控物理资源 CPU、存储器、交换、磁盘读写和网络读写的使用状况，应用程序监视器负责检测应用程序的状态。

瘦运行环境有三个组成部分：管理者、调度器、Web 门户。管理者负责处理用户的请求，管理资源，与公共服务框架交互。调度器负责调度作业或分发请求。Web 门户是一个图形用户界面，最终用户通过它提交和监视作业或应用程序。

图 2 显示了两个瘦运行环境和公共服务框架之间的交互。两个服务提供商运行异构负载

4 Parallel Batch Job，并行批处理作业

的协调和资源提供商的情况如下：

资源供应服务商制订的资源供应政策决定什么时间将多少资源提供给瘦运行环境，如何协调两个协同的运行时环境之间的资源。第三章中，我们介绍了资源供应政策。

管理器即每个瘦运行环境的管理实体，其上的资源管理策略决定了管理器请求或释放资源的时机和数量，按照什么样的政策提供资源供应服务。

对于不同的负载，调度策略有不同的含义。对于并行批处理作业或 MapReduce 作业，调度策略决定调度器何时以及如何选择工作（或任务）运行。对于 Web 服务器，调度策略包括两个具体的政策：实例调整政策和请求分配政策。实例调整政策决定 Web 服务器的实例被调整的时机和程度，请求分配政策决定根据什么标准如何分发请求。

以一个 web 服务瘦运行环境和一个并行作业瘦运行环境分别与公共服务框架交互为例，过程如下：

(1). web 服务管理器从资源供应服务获得资源下限内的初始资源，并以相匹配的规模运行 Web 服务器实例。

(2). web 服务管理器和负载均衡器（一种负责为 Web 服务器实例分配负载的调度器）交互，以设置其请求分配策略。web 服务管理器向负载平衡器注册网络服务器实例的 IP 和端口信息，负载平衡器根据请求分配政策分发请求到 Web 服务器实例。我们整合 Linux 虚拟服务（LVS）(http://www.linuxvirtualserver.org/) 作为 IP 层的负载平衡器。

(3). 在每个节点上的监视器定期检查资源的利用率并报告给 web 服务管理器。如果性能值超过阈值，例如，平均实例消耗的 CPU 使用率超过 80％，web 服务管理器会根据实例调整政策调整 Web 服务器实例的数量。

(4). 根据当前的 Web 服务器实例，web 服务管理器向资源供应服务请求或释放资源。

对于并行批处理作业或 MapReduce 作业，PhoenixCloud 采用资源管理政策如下：

租赁的资源单位时间用 L 表示。我们假定资源的租期是租赁资源的单位时间的正整数倍。在类 EC2 系统中，对于并行批处理作业，每个最终用户负责手工管理资源。我们假定如果一个作业运行结束，用户只在每个租赁资源单位时间的末尾释放资源。这是因为：第一，类 EC2 系统以租赁资源的时间单位（一小时）收取资源使用费用；第二，最终用户很难预测作业完成时间，因此即时释放资源给资源提供商几乎是不可能的。

我们定义“资源调整比例”为，在队列中的所有作业所积累的资源需求与瘦运行环境所拥有的现有资源之比。当资源调整比例大于 1 时，说明对于需要立即运行的作业而言，必须拥有比目前瘦运行环境已有的资源更多的资源才能满足其运行需求。

我们设置了两个调整资源的阈值，分别为“请求资源的比例阈值”（用 U 表示）和“释放资源的比例阈值”（用 V 表示）。请求和释放资源的过程如下：

(1). 并行批处理作业管理器注册一个周期性定时器（租赁资源的时间单位），触发每个单位时间的资源调整。在定时器的推动下，并行批处理作业管理器周期性地扫描队列中的作业。

(2). 如果调整资源比例超过了请求资源比例的阈值，并行批处理作业管理器会请求大小

为 DR1 的资源，DR1 计算方法如下：

对并行批处理作业：

DR1 =（(队列中的所有作业需要资源的总和）–（当前并行批处理作业瘦运行环境拥有的资源））/ 2）

对 MapReduce 作业：

DR1 =目前队列中最大的作业所需的资源

(3). 对并行批处理作业，如果资源调整比例不超过正在请求资源比例的阈值，但目前在队列中最大作业的资源需求的比例比瘦运行环境所拥有的资源的比例大， 并行批处理作业管理器将按照 DR2 的大小请求资源，DR2 计算方法如下：

DR2=（目前队列中最大的作业所需要的资源）-（目前由瘦运行环境拥有的闲置资源）

对于并行批处理作业，当目前在队列中最大作业的资源需求比例比瘦运行环境所拥有的资源比例大时，这意味着最大的作业由于没有可利用的资源，将无法运行。请注意，明显不同于并行批处理作业，MapReduce 的任务是相互独立的[28]，所以不会发生这种情况。

(4). 如果资源调整比例比释放资源比例阈值低， 并行批处理作业管理器将按照 RSS（ReleaSing Size）的大小释放闲置资源。其计算方法为：

RSS =（并行批处理作业瘦运行环境所拥有的闲置资源）/2。

(5). 如果资源供应服务主动提供资源给并行批处理作业管理器，后者将获得资源。

郑等人[22]认为，在管理数据中心 Web 服务时，实际的实验比许多管理任务模型更便宜，更简单，更准确。我们也持同样的看法。在本文中，我们通过部署的实际系统来决定两台 Web 服务器和一个搜索引擎负载跟踪的资源管理政策。

5 性能评价

5.1 真实负载日志

对于并行批处理作业，我们从 http://www.cs.huji.ac.il/labs/parallel/workload/选择三个典型负载日志：美国宇航局（NASA）的 iPSC，SDSC BLUE 和 LLNL Thunder。美国宇航局 iPSC 是为期两个星期的真实的日志片段，从 1993 年 10 月 01 日 0 时 00 分 03 秒（PDT[5]）开始，对应的集群系统配置为 128 个节点。 SDSC BlUE 是为期两个星期的真实日志片段，从 2000 年 04 月 25 日 15 点 00 分 03 秒(PDT)开始，对应的群集配置为 144 节点。LLNL Thunder 是一个为期两个星期的真实的日志片段，从 2007 年 2 月 1 日 18 时 10 分 25 秒(段)开始，对应的集群系统配置是 1002 节点（不包括管理节点等）。

对于 Web 服务器，我们使用从 http://ita.ee.lbl.gov/html/traces.html 获得的两个真实的负载日志：一个是世界杯期间（记为 WorldCup）的负载日志，时间为从 1998 年 6 月 7 日至 6 月 20 日[2]，另外一个是从繁忙的互联网服务供应商 ClarkNet 获得的 HTTP 负载日志，时间从 1995 年 8 月 28 日至 9 月 10 日。同时，我们选择公开可用的匿名搜索引擎的负载日志，

[5] 太平洋夏令时（适用于美国西部加州、华盛顿州、俄勒冈州等）

时间在 2011 年从 3 月 28 日零点 00 分 00 秒到 23:59:59[54]，我们在本文的其余部分称之为搜索负载（SEARCH）。

5.2 合成负载日志

据我们所知，在同一平台上共存的并行批处理作业、Web 服务器、搜索引擎、MapReduce 作业负载日志无法公开获得。因此，在我们的实验中，根据 5.1 节介绍的实际负载日志，我们创建了合成的负载日志。我们使用二元组$\langle \boldsymbol{PRC_A}, \boldsymbol{PRC_B} \rangle$[6]代表两个异构负载日志：A 具有较低的优先级，例如，并行批处理作业或 MapReduce 作业，而 B 具有更高的优先级，例如，Web 服务器或搜索引擎，$\boldsymbol{PRC_A}$ 是 A 中一个最大的作业的最大的资源需求，$\boldsymbol{PRC_B}$ 是 B 的尖峰资源消耗，例如，二元组$\langle 100, 60 \rangle$，是在 SDSC BLUE 和 World Cup 的跟踪日志的基础上得来的，有两方面的含义：（1）. 我们按照不同的缩放常量因子分别缩放了 SDSC BLUE 和 World Cup 负载；（2）.在相同的仿真集群系统，在 SDSC BLUE 中最大的作业的最大资源需求和 World Cup 的尖峰资源消耗分别是 100 和 60 个节点。

5.3 实验方法

为评价和比较 PhoenixCloud 和类似 EC2+ RightScale 的系统，我们采取以下实验方法。

5.3.1 实际系统上部署的实验

对于 Web 服务器和搜索引擎负载，通过在真实系统上部署和运行 5.1 节提到的三个实际负载，我们获得资源消耗日志。对于 MapReduce 作业，我们也通过在物理硬件上部署和运行实际系统进行实验，硬件配置如 5.5.2 节所示。

5.3.2 为异构负载设计的模拟实验

一个典型的负载日志的周期往往是几周，甚至几个月的时间。为评价系统中许多关键因素对实验结果的影响，我们需要经常执行耗时的实验。然而，当我们进行几百种负载日志实验时，它们的资源消耗多达上万个节点，采用真实系统实验的方法明显是不值得的。所以我们使用模拟方法加速实验进程。我们设置加快提交和完成的作业的因子为 100。这种加速使得在三个小时左右能完成两个星期的日志重放。此外，在第 5.7 节，我们在物理硬件上部署了真实系统来验证我们模拟系统的精度。

5.4 测试床

如图 3 所示，测试平台包括三种类型的节点，名称分别以 glnode，ganode，gdnode 开头，每个 glnode 节点配置为 2GB 内存和两个四核英特尔至强的（2.00GHz）处理器，操作系统是一个 64 位的 2.6.18-xen 内核的 Linux。每个 ganode 节点配置为 1 GB 的内存和 2 个 AMD Optero242（1.6GHz）的处理器，操作系统是 64 位的 2.6.5-7.97-SMP 版本内核 Linux。每个 gdnode 类节点配置为 4GB 的内存和一个四核英特尔至强 1.60GHz 处理器，操作系统是一个 64 位的 2.6.18-194.8.1.el5 版本内核的 Linux。所有节点都

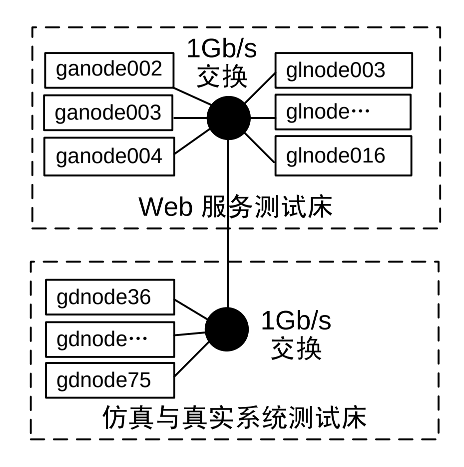


图3.　测试床

[6] PRC, Peak Resources Consumption, 峰值资源消耗

使用 1Gb/s 的交换机连接。

5.5 实际系统上部署的实验

5.5.1 运行真实网络服务器和搜索引擎系统

在这一节，我们部署真实的系统，以获取两个 Web 服务器和一个搜索引擎负载的资源消耗跟踪日志。

我们在 glnode 类的每个节点上部署了 8 个 Xen 虚拟机（http://www.xensource.com/）。对于每个 Xen 虚拟机，分配单核心处理器和 256 MB 内存，使用 64 位的 2.6.18XEN 内核版本 CentOS 操作系统。我们部署了真实的 PhoenixCloud 系统：负载平衡器是直接路由模式的 LVS（HTTP：// kb.linuxvirtualserver.org/wiki/ LVS/ DR）；每个虚拟机上分别部署了一个代理和一个监视器，由于负载很轻，因此 LVS 和其他服务都部署在 ganode004 节点上。

我们选择最少连接调度策略（http://kb.linuxvirtualserver.org/wiki/Least-Connection Scheduling）来分发请求，选择 httperf（http://www.hpl.hp.com/research/的 Linux/httperf/）作为负载发生器，开源应用程序 ZAP！（http://www.indexdata.dk/）作为目标 Web 服务器。httperf LVS 和 ZAP！的版本分别是 0.9.0，1.24 和 1.4.5。两个 httperf 实例部署在 ganode002 和 ganode003 上。

Web 服务器的负载日志是由 WorldCup 负载[2]使用 2.22 缩放因子得到。实验包括两个步骤。第一步，我们通过实验获得决定实例调整政策所需的数据；第二步，获得资源消耗日志。

在第一步中，我们部署的 PhoenixCloud 禁用了实例调整政策。在这个配置下，web 服务管理器不会调整 Web 服务实例的数量。在 16 个虚拟实验的测试床上部署了 16 个 ZAP！实例，即每个虚拟机上部署一个实例。当 httperf 生成不同规模的负载时，我们记录了实际的吞吐量，平均响应时间和 CPU 核心的平均使用率。因为我们保证一个 CPU 核心被分配到一台虚拟机上，对于虚拟机来说，VCPU 的数量就是 CPU 的核数。因此，每个 CPU 核心的平均使用率就是 VCPU 的平均利用率。我们观察到，当 VCPU 的平均使用率低于 80％时，请求的平均响应时间小于 50 毫秒。然而，当 VCPU 的平均利用率增加为 97％，请求的平均响应时间大幅增加至 1528 毫秒。

基于上述观察形成了我们的实例调整政策，我们选择用 VCPU 的平均利用率为准绳来调整 ZAP！实例数，并设置 80％作为阈值。对于 ZAP！，我们制订的实例调整政策如下：初始服务实例有两个。如果在过去的 20 秒，所有的 Web 服务实例所消耗 VCPU 的平均利用率超过 80％，web 服务管理器将增加一个实例。如果当前所有的 Web 服务实例所消耗 VCPU 的平均利用率小于过去的 20 秒的 $(0.80\times((\boldsymbol{n}-1)/\boldsymbol{n}))$（其中 $\boldsymbol{n}$ 为当前实例的数量），web 服务管理器将减少一个实例。

在第二步，我们部署 PhoenixCloud 时启用了实例调整政策。web 服务管理器根据实例调整政策调整 Web 服务实例的数量。在实验中，我们仍然记录实际吞吐量、平均响应时间和虚拟机的数量之间的关系。我们观察到，平均响应时间低于 700 毫秒时，当虚拟机数量增加，吞吐量呈线性增加。这表明，实例调整政策是合适的，虽然可能并不是最优的。

在上述策略中，我们获得了 WorldCup 负载跟踪两个星期资源消耗日志。用同样的方法，我们分别使用 772.03 和 171.29 为缩放因子获得两个星期 ClarkNet 负载和一个匿名搜索引擎——SEARCH[54]的资源消耗跟踪日志。资源消耗的追踪日志，可以在附录 B.1 中查看。

5.5.2 在硬件上执行 MapReduce 作业的实验

本节通过在真实系统上部署 MapReduce 作业负载评价了 PhoenixCloud 系统。

测试平台由 20 个 X86-64 节点组成，即图 3 中，编号从 gdnode36 到 gdnode55 的节点。我们选择了一系列广泛使用的 MapReduce 作业作为我们的 MapReduce 负载，其中的细节，可以在附录 B.5 中查看。同时，我们重放在 5.5.1 小节介绍的 SEARCH 资源消耗跟踪日志。我们使用 $\langle \boldsymbol{PRC_A}, \boldsymbol{PRC_B} \rangle$ 表示 MapReduce 作业和搜索引擎的负载的尖峰资源消耗。在这个实验中 $\langle \boldsymbol{PRC_A}, \boldsymbol{PRC_B} \rangle$ 是 $\langle 19, 64 \rangle$。

表1. MapReduce作业和搜索引擎负载—SREACH的资源提供商度量

系统	峰值资源消耗	资源消耗(节点数×小时)	节约的峰值消耗	节约的资源消耗
PhoenixCloud	78	960	63.21%	57.33%
非协作	212	2250	0	0

实验中使用 0.21.0 版本的 Hadoop 运行 MapReduce 作业，Hadoop 的 JDK 版本是 1.6.0.20。我们在一个节点上配置 Namenode（名字节点）和 jobtracker（作业跟踪器），而在另外一个节点上配置Datanode（数据节点）和TaskTracker（任务跟踪器）。在Hadoop中，块大小为64MB，在每个节点上共有4个映射计算槽和2个规约计算槽，并且调度策略为先来先服务（FCFS[7]）。

表2. MapReduce作业的服务提供商指标

系统	完成作业数	平均周转时间（秒）	资源消耗(节点数×小时)
PhoenixCloud	477	568	960
非协作	477	310	2250

我们采取在第 4 章介绍的资源管理政策。在 PhoenixCloud 系统中，对于 MapReduce 作业，基线参数是[R0.28/E19/U5/V0.05/L60][8]。我们分别设置资源预订阈值（RBT）和使用率参考值（URR）的值为 1。在 Hadoop 中，一个 MapReduce 作业的资源需求由它的 map 任务的数量来决定，因此请求资源数为 map 任务数除以每个节点的 map 计算槽数。为防止 MapReduce 作业尖峰资源需求超过测试平台的系统容量，我们使用缩放比例来缩放每个资源请求（即资源请求除以缩放比例）。对于目前的 20 节点测试平台，缩放比例为 16。对于类 EC2 系统，我们根据其时间戳一个接一个运行 MapReduce 作业。最后，我们累加所有 MapReduce 作业的资源消耗获得其峰值和资源消耗总量。出于同样的原因，我们还是使用缩放因子，以减少 MapReduce 作业的尖峰资源需求。

从上面的结果可以看到，我们的算法很好地适用于 MapReduce 作业。相比非协同的类 EC2+RightScale 系统，PhoenixCloud 可以分别降低 63.21％的尖峰资源消耗和 57.33％资源消耗总量。与此同时，PhoenixCloud 系统的吞吐量和类 EC2 系统相同，在 PhoenixCloud 和类 EC2 系统中，每个作业的平均周转时间分别为 568，310 秒，平均每个作业延迟 258 秒。

5.6 模拟实验

在本节中，我们对类 EC2+RightScale 和 PhoenixCloud 系统使用 Web 服务器和并行批处理作业负载的追踪进行了性能比较。实验设置如下：

5.6.1 模拟系统

(1). **模拟集群** 上文提到的负载跟踪是从不同配置的平台获得的。例如，美国宇航局 iPSC 是从由单处理器节点组成的集群系统上获得的；SDSC BLUE 是从由 8 个处理

[7] First Come First Served
[8] B 代表协同资源的大小（节点数）；U 代表请求资源的比例阈值；V 代表释放资源的比例阈值；G 代表释放资源弹性系数；L 代表租赁时间单位。

器节点组成的集群系统获得的；LLNL Thunder 是从由处理器节点组成的集群系统上获得的。在其余的实验中，我们的模拟集群系统以美国宇航局 iPSC 的集群为蓝本，只由单处理器节点组成。

(2). **模拟类 EC2+RightScale 系统** 基于 PhoenixCloud 框架，我们实现和部署了类 EC2+ RightScale 系统的测试平台，如图 4 所示。模拟系统包括两个模拟模块：工作模拟器和资源模拟器。相比真实的 PhoenixCloud 系统，我们只保持资源供应服务，web 服务管理器和并行批处理作业管理器。作业模拟器读取节点上的跟踪日志文件并发送资源请求给资源供应服务，资源供应服务为每个作业分配相应的资源。当每个作业运行完成，作业模拟器将释放资源给资源供应服务。资源模拟器模拟资源消耗的变化并驱动 web 服务管理器请求或释放资源给资源供应服务。因为 RightScale 对于 Web 服务负载提供与 PhoenixCloud 系统相同的可伸缩管理，我们只是使用了第 5.5.1 节提到的相同的 Web 服务资源消耗追踪日志。

(3). **模拟 PhoenixCloud** 我们模拟了图 2 中所示的真实 PhoenixCloud 系统。其中保持了资源供应服务、并行批处理作业管理器、web 服务管理和调度器，其他的服务被删除了。资源供应服务执行第 3 章定义的协同的资源供应模型。

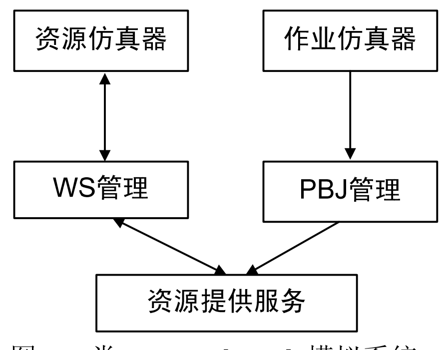


图4.　类EC2+RightScale模拟系统

5.6.2 实验配置

(1). **并行批处理作业调度策略** 对并行批处理作业，PhoenixCloud 采取首次适应（first-fit）调度策略。首次适应调度策略以作业到达的顺序扫描队列中所有的作业，系统会选择能够满足其资源需求的第一个作业去执行。类 EC2 系统不需要调度策略，因为并行批处理作业运行时，每个最终用户负责手动请求或释放资源。

(2). **资源管理政策** 对于并行批处理作业，我们采用了第四章描述的资源管理政策。

5.6.3 系统极限容量下的实验

在附录 B.2 中，我们假定资源提供商有无限的资源，并展示了实验结果。在本节中，我们假设的资源提供商有不同的系统容量的有限资源，然后我们报告了实验的结果。

5.1 节中介绍的六个负载日志有三种组合。我们使用如下异构负载组合：（（IPSC，WorldCup），（SDSC，Clark），（LLNL，SEARCH）），它们的资源数量分别是：（（128，128）（144，128），（1002，896））。

通过大量的实验比较，我们设置 PhoenixCloud 系统的基线参数：（IPSC, WorldCup）和（SDSC，Clark）为[R0.5/E400/U1.2/V0.1/L60]，（LLNL,SEARCH）为[R0.5/E3000/U1.2/V0.1/L60]。 [Ri/Ej/Uk/Vl/Lm]表示: R 代表两个资源下限的总和与 PRC_A 和 PRC_B 的总和的比值，值为 i。E 表示两个资源的上限边界的总和，值为 j；U 代表请求资源比例阈值，值为 k；资源释放阈值比例用 V 表示，l 为值.，租赁资源的时间单位用 L 表示，m（分钟）为值。在附录 B.4 节，我们探讨了 PhoenixCloud 系统中不同参数的影响。

每个实验进行 6 次，我们的报告取 6 次实验的平均值。在表 3 中，对于不同的系统的容量，即 4000、5000、6000，除了服务提供商的基线参数，我们将使用率参考值 URR 和资源预定阈值 RBT 分别设置为 0.5 和 2。对于 ***E***（上面提到的两个资源上界的总和），E[A/B/C] 分别表示:对于（IPSC，WorldCup）负载组合,两个上限的总和是 A；对于（SDSC，Clark）负载组合，两个上限的总和为 ***B***；对于（LLNL，SEARCH）负载组合，两上界的总和是 ***C***。

表3. 不同系统容量的详细配置

系统容量（节点数）	系统	E[A/B/C]	URR†	RBT※
4000	PhoenixCloud	[400/400/3000]	0.5	2
5000	PhoenixCloud	[400/400/3000]	0.5	2
6000	PhoenixCloud	[400/400/3000]	0.5	2

† 使用率参考值，Usage rate reference
※ 资源预定阈值，Resource booking threshold

从表 4、表 5、表 6、表 7 和表 8，我们可以看出，对于不同系统的容量配置,我们的算法性能特别好：

从资源提供商角度看，Phoenix Cloud 可以节省相当数量的尖峰资源消耗。当 PhoenixCloud 的系统容量下降到 4000 个节点时，仅仅是类 EC2+RightScale 容许的最小系统容量的 65.3%，PhoenixCloud 可以分别比类 EC2+RightScale 系统节省尖峰资源消耗和资源消耗总量 20%和 43%。这有两方面原因：（1）.与在类 EC2 系统中每个批处理作业所需的资源将会立即满足不同，并行批处理作业可在 PhoenixCloud 系统中排队，因此资源供应商使用 PhoenixCloud 系统可以降低峰值资源消耗和资源消耗总量;（2）.在 PhoenixCloud 系统中，在竞争条件下设置使用率参考值 URR 和比例共享因子 PSF 可以有效地降低尖峰资源消耗。

从服务提供商角度看，以完成工作的数量计，PhoenixCloud 吞吐量与类 EC2+RightScale 系统变化不大（最大跌幅仅为 0.2%）；平均周转时间有少量的增加（对 NASA 的 iPSC、SDSC 和 LLNL，最大延迟时间增加分别为为 34%、34%和 21%）而相应的资源消耗量分别比类 EC2+RightScale 系统节省 35%、13%和 21%。究其原因如下：并行批处理作业可以在 PhoenixCloud 排队，而在类 EC2 系统中每个批处理作业所需的资源将会立即满足，因此作业完成的数量和平均周转时间反而会受到影响。PhoenixCloud 的资源管理策略对性能的影响很小，但是服务提供商的资源消耗却能显著节省。

表4. PhoenixCloud系统不同系统容量下6个真实负载对应的资源提供商指标

系统容量	系统	峰值资源消耗 (节点数)	有效资源消耗 (节点数×小时)	资源消耗总量 (节点数×小时)
六个负载	非协作	6126	677190	687441
4000	PhoenixCloud	3518	545450	551845
5000	PhoenixCloud	3283	546395	552846
6000	PhoenixCloud	2899	543569	549832

表5. NASA iPSC负载在系统不同容量下的服务提供商指标

系统容量	系统	峰值资源消耗 (节点数)	平均周转时间 (秒)	资源消耗 (节点数×小时)
六个负载	非协作	2603	573	54118
4000	PhoenixCloud	2603	766	35786
5000	PhoenixCloud	2603	738	35621
6000	PhoenixCloud	2603	733	35146

表6. SDSC Blue 负载在不同系统容量下的服务提供商指标

系统容量	系统	完成作业数	平均周转时间(秒)	资源消耗(节点数×小时)
六个负载	非协作	2657	1975	35838
4000	PhoenixCloud	2656	2642	31231
5000	PhoenixCloud	2652	2546	31654
6000	PhoenixCloud	2654	2535	31573

表7. LLNL Thunder负载在不同系统容量下的服务提供商指标

系统容量	系统	完成作业数	平均周转时间(秒)	资源消耗(节点数×小时)
六个负载	非协作	7273	2465	339416
4000	PhoenixCloud	7261	2941	270847
5000	PhoenixCloud	7262	2882	271233
6000	PhoenixCloud	7262	2978	268298

表8. PhoenixCloud系统在不同系统容量下所节省的资源使用峰值和资源消耗总量

系统容量	系统	节约的峰值资源消耗	节约的资源消耗总量
4000	PhoenixCloud	42.57%	19.72%
5000	PhoenixCloud	46.41%	19.58%
6000	PhoenixCloud	52.68%	20.02%

5.6.4 不同数量合成异构负载的实验

本节我们评估 PhoenixCloud 在从 6 到 120 个不同的合成异构负载组合下的效率。

合成的负载是基于 5.1 节中的介绍生成的，分别是（IPSC,WorldCup）、（SDSC，Clark）、（LLNL，SEARCH）。当我们执行一组（6N + 6）个负载的跟踪实验，我们产生 N 个新的负载组合如下：

我们记录第 $\boldsymbol{i}$ 个负载组合为：

$$((\boldsymbol{iPSC_i}, \boldsymbol{WorldCup_i}), (\boldsymbol{SDSC_i}, \boldsymbol{Clark_i}), (\boldsymbol{LLNL_i}, \boldsymbol{SEARCH_i}), \boldsymbol{i}=1,\cdots,\boldsymbol{N})。$$

把 $((\boldsymbol{iPSC},\boldsymbol{WorldCup}),(\boldsymbol{SDSC},\boldsymbol{Clark}),(\boldsymbol{LLNL},\boldsymbol{SEARCH}))$ 每个负载日志分成（$\boldsymbol{N}$ +1）部分。在相同负载日志组合中，为防止在同一时期创建重复的跟踪负载，对于 ***iPSC***，***SDSC*** 或 ***LLNL***，我们交换前 $\boldsymbol{i}$ 部分和其余部分，获得

$$((\boldsymbol{iPSC_i}),(\boldsymbol{SDSC_i}),(\boldsymbol{LLNL_i}))；$$

对于 ***Clark***，***WorldCup*** 或 ***SEARCH***，我们交换后 $\boldsymbol{i}$ 部分和其余部分获得

$$((\boldsymbol{WorldCup_i}),(\boldsymbol{Clark_i}),(\boldsymbol{SEARCH_i}))。$$

对并行批处理作业负载的跟踪，我们采取 ***IPSCi*** 作为一个例子：首先，我们定义 $\boldsymbol{Sub}=((336\times3600\times\boldsymbol{i})/(\boldsymbol{N}+1))$ 第二，对于已经提交的作业，如果其时间戳 ***t*** 比 ***Sub*** 小，我们重置其提交时间为（***t*** +（336×3600 -Sub）)，否则为 ***t-Sub***。

对于 Web 服务负载的跟踪，我们以 $\boldsymbol{Clark_i}$ 作为一个例子。我们定义 $\boldsymbol{MP}=((336\times\boldsymbol{i})/(\boldsymbol{N}+1))$。$\boldsymbol{RC_j},\boldsymbol{j}=1,\cdots,336$ 是 ***Clark*** 负载跟踪的第 ***j*** 个小时的资源消耗。对于合成负载 $\boldsymbol{Clark_i},\boldsymbol{i}=1,\cdots,\boldsymbol{N}$，如果 $\boldsymbol{j}<\boldsymbol{MP}$，$\boldsymbol{RC_{i,j}}=\boldsymbol{RC_{336-MP+i}}$，否则，$\boldsymbol{RC_{i,j}}=\boldsymbol{RC_i}-\boldsymbol{MP}$。

由于使用了大量的负载跟踪，我们假定资源提供商有无限的资源，并分别设置资源预定阈值和参考使用率为 1（解释见第三章）。我们在 PhoenixCloud 系统中设置基线参数：[R0.5/E100000/U1.2/V0.1/L60]。E100000 意味着每个服务提供商有更大的资源上限，并会得到足够多的资源。表 9 总结了类 EC2+RightScale 系统和 PhoenixCloud 系统运行不同数量的负载在资源提供商维度上的指标比较。

从表 9 中，我们可以观察到更大规模的部署，我们的解决方案可以节省更多的服务器成本。对于小规模（6126 节点）、中等规模（28206 节点）、大规模部署（116030 节点），我们的解决方案分别比类 EC2+RightScale 方案节省服务器成本 18.01％、57.60％和 65.54％。

表9. 运行不同数目负载时,PhoenixCloud系统所节省的峰值资源和资源消耗总量

工作负载数目	峰值资源消耗(类 EC2+RS†系统)	节约的峰值资源消耗	节约的资源消耗总量
6	6126	18.01%	17.22%
30	28206	57.60%	15.32%
60	56920	61.14%	16.52%
120	116030	65.54%	16.56%

† RightScale

请注意，表 9 中，对于六个负载的跟踪，PhoenixCloud 系统的配置与表 8 略有不同，因此有不同的结果。首先，在表 9 中，我们假定每个服务提供商有一个较大的资源上限；第二，表 9 中，我们假定资源提供商拥有无限的资源，将资源预定阈值和参考使用率分别设置为 1 。

5.7 模拟系统的准确性

为了验证我们模拟系统的精度，我们在测试平台上部署了真正PhoenixCloud系统。测试平台由40个X86-64节点组成，如图3所示，编号从gdnode36到gdnode75。

真实的Web服务负载已经在5.5.1节运行过了，因此此处我们在物理硬件上运行并行批处理负载。我们合成了一个并行批处理作业的负载日志，其中包括大小从8到64个CPU不等的100个作业。100个作业需要在10小时内提交到PhoenixCloud系统，平均提交作业的时间间隔为300秒。我们的实验包括两个步骤：首先，我们把合成的并行批处理负载提交到真实的PhoenixCloud系统上，然后收集负载日志。第二，我们根据通过实际系统的实验获得的负载日志，在模拟系统上再次提交了相同的负载，然后比较真实和模拟系统的指标，以评估模拟系统的准确性。PhoenixCloud系统的基线参数是 [R0.5/E40/U1.2/V0.1/L60]，我们分别设置资源预订阈值RBT和使用率参考值URR为 1 。通过实验，我们发现，真实PhoenixCloud系统的尖峰资源消耗和资源消耗总量的比值和模拟系统的都分别约为1.14，由此验证，我们的模拟系统是足够准确的。

6 相关工作

国内外相关工作可以从以下四个角度总结：

6.1 节省数据中心成本的方法和系统

基于虚拟化的聚集方面，沃格尔斯（Vogels）等[50]总结了过去的和最新的工作，提出了基于虚拟化的聚集方案，以应对应用程序隔离或操作系统异构所造成服务器扩张。在托管数据中心中,很多人利用虚拟机器为多层次的服务或高性能计算[20]提供弹性资源供应[49] [48]。切斯（Chase）等人[43]提出在托管数据中心为为共存的服务供应资源，自动适应负载，从而提高服务器集群的能源效率。

统计复用方面， 萨哈里亚（Zaharia）等[27]指出，过去的EC2系统利用统计复用技术来降低服务器的购置成本。乌尔噶昂卡（Urgaonkar）等人[39]提出的超售资源的方法，最大限度地为更多的大量轻负载应用程序提供资源,使得这些应用程序可以在一个给定的系统配置上运行，但是他们只使用大量固定强度的轻负载服务来验证结果。经过对多个单一的或合并的应用程序做详细的分析，崔（音译， Choi）等[53]为预测聚集的应用的平均功耗和维持功耗开发了相应的模型。

6.2 数据中心软件基础设施

OpenNebula（http://www.opennebula.org/）和Haizea（http://haizea.cs.uchicago.edu/）这两个相辅相成的开源项目可以用来管理虚拟私有/混合云的基础设施[25]。斯坦纳（Steinder）等人[18]的研究表明，虚拟机器使得异构负载可以构建在任何服务器上。我们以前研制的DawningCloud系统[20] [21]旨在为解决科研社区如何从云计算的规模经济中受益的问题。辛德曼（Hindman）等人[26]开发了Mesos系统——一个多种集群的计算框架，包括：Hadoop和MPI等。在[5][18][25][20][26]中提到的数据中心软件基础设施系统都属于同一领域的研究工作，但是这些均不支持异构负载的协同资源供应。

6.3 资源供应和调度

斯坦纳等[18]针对同构负载系统，开发了一系列新的自动化机制，从而可以更有效地调度非交互式作业负载。西尔韦斯特(Silberstein)等[16]为具有不同资源需求的大规模并行任务设计了一个相应的调度算法。

林斌（音译， Lin）等[12]提供了一个系统级调度机制，VSched。VSched可以保证交互式负载的执行率和交互性能，并在一台服务器上为虚拟机提供软实时保证。马戈（Margo）等[13]研究TeraGrid系统的元调度能力（网格应用的联合调度），包括分布式集群格点之间的用户资源预留设置。索托马约尔（Sotomayor）等[17]提出了租赁管理架构的设计，但只考虑了同构负载，即混合最佳适应租赁请求和提前预订请求的并行批处理作业。

艾萨德（Isard）等[28]和萨哈里亚等[27]的研究重点在于解决直接连接存储的低端系统上类MapReduce数据密集型作业的调度公平性和数据局部性之间的冲突。在异构的主从平台上，贝诺特（Benoit）等[54]关注于处理独立和相同任务的集合组成的多应用的调度问题。萨德哈希瓦姆（Sadhasivam）等[55]提出以更细粒度执行队列中的作业，以增强 Hadoop中调度的公平性。

在非协同资源供应和调度方面，瓦尔斯普尔热（Waldspurger）等[57]实现了彩票调度——一个新的随机资源分配机制，它对计算相对的执行率提供了有效的应答式的控制。金伟（译音， Jin）等[58]提出了一种以中间人请求调度的方式实现比例共享资源控制的方法，来共享服务。

6.4 数据中心和云基准测试程序

基准评估程序是评价计算机系统的基础。詹剑锋等人[39]系统地确定了数据中心三类以吞吐量为导向的负载，其目标是增加以处理的用户请求、数据、支持的最大并发用户数来计量的吞吐量（Volume of throughput）；同时为描述负载和为负载而设计的数据中心系统定义了一个新术语-高通量计算（High Volume Throughput Computing, HVC）。罗纯杰（Luo）等[59]提出一个新的测试基准程序集，CloudRank-D，用来评测和比较运行大数据应用程序的云计算系统。贾（Jia）等[38]选取了不同领域的二十一个代表性应用作为大数据应用测试程序基准集—BigDataBench。

7 结束语

在本文中，我们通过充分利用异构负载的差异性，提出了协同资源供应解决方案，以节省服务器购置成本。为此，我们构建了一个创新性的系统，称为PhoenixCloud系统。此系统能够为以下四种有代表性的异构负载提供协同供应资源：并行批处理作业、Web服务器、搜索引擎、MapReduce作业。我们提出一个新的实验方法，将真实和模拟系统的实验相结合，并对我们的系统进行了综合评价。我们的实验显示，对于非协同的类EC2+RightScale系统，通过在更大的粒度（两种异构负载的组合）上利用统计复用的优势，我们的算法获得了很大收益；对于小规模（6126节点）、中等规模（28206节点）、大规模部署（116030节点），我们的解决方案比非协同资源供应系统分别节省服务器成本18%、58%、66％。

参考文献：

[1] K. Appleby et al. 2001. Oce´ ano–SLA Based Managemen of a Computing Utility. in Proc. of IM 2001, pp. 855-868.

[2] M. Arlitt et al. 1999. Workload Characterization of the 1998 World Cup Web Site, Hewlett-Packard Company, 1999

[3] A. AuYoung et al. 2006. Service contracts and aggregate utility functions. in Proc. Of 15th HPDC, pp.119-131.

[4] A. Bavier et al. 2004. Operating system support for planetary-scale network services. in Proceedings of NSDI 04, pp. 19-19.

[5] J. S. Chase et al. 2001. Managing energy and server resources in hosting centers. in Proc. of SOSP ’01, pp.103-116.

[6] J. S. Chase et al. 2003. Dynamic Virtual Clusters in a Grid Site Manager. in Proc. of the 12th HPDC, pp.90-103.

[7] A. Fox et al. 1997. Cluster-based scalable network services. SIGOPS Oper. Syst. Rev. 31, 5 (Dec. 1997), pp. 78-91.

[8] K. Gaj et al. 2002. Performance Evaluation of Selected Job Management Systems. in Proc. of 16th IPDPS, pp.260-260.

[9] L. Grit et al. 2008. Weighted fair sharing for dynamic virtual clusters. in Proceedings of SIGMETRICS 2008, pp. 461-462.

[10] K. Hwang et al. Scalable Parallel Computing: Technology, Architecture, Programming, and McGraw-Hill 1998.

[11] D. Irwin et al. 2006. Sharing networked resources with brokered leases. in Proc. of USENIX’06, pp.18-18.

[12] B. Lin et al. 2005. Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. in Proc. of SC 05, pp.8-8.

[13] M. W. Margo et al. 2007. Impact of Reservations on Production Job Scheduling. in Proc. of JSSPP 07, pp.116-131.

[14] B. Rochwerger et al. 2009. The Reservoir model and architecture for open federated cloud computing IBM J. Res. Dev., Vol. 53, No.4,2009.

[15] P. Ruth et al. 2005. VioCluster: Virtualization for dynamic computational domains. in Proceedings of Cluster 05, pp.1-10.

[16] M. Silberstein et al. 2006. Scheduling Mixed Workloads in Multigrids: The Grid Execution Hierarchy. in Proc. of 15th HPDC, pp.291-302.

[17] B. Sotomayor et al. 2008. Combining Batch Execution and Leasing Using Virtual Machines. in Proc. of HPDC 2008, pp.87-96.

[18] M. Steinder et al. 2008. Server virtualization in autonomic management of heterogeneous workloads. SIGOPS Oper. Syst. Rev. 42,1 (Jan. 2008), pp.94-95.

[19] J. Zhan et al. 2005. Fire Phoenix Cluster Operating System Kernel and its Evaluation. in Proc. of Cluster 05, pp.1-9.

[20] L. Wang et al. 2009. in cloud, do MTC or HTC service providers benefit from the economies of scale?. in Proc. of SC-MTAGS ’09. ACM, New York, NY, 1-10.

[21] L. Wang et al. 2011. in Cloud, Can Scientific Communities Benefit from the Economies of Scale?. IEEE TPDS. vol.23, no.2, pp.296-303, Feb. 2012

[22] W. Zheng et al. 2009. JustRunIt: Experiment-based management of virtualized data centers, in Proc. of USENIX 09.

[23] Z. Zhang et al. 2006. Easy and reliable cluster management: the self-management experience of Fire Phoenix, in Proc. of IPDPS 2006.

[24] J. Zhan et al. 2008. Phoenix Cloud: Consolidating Different Computing Loads on Shared Cluster System for Large Organization. in Proc. of CCA’ 08. http://arxiv.org/abs/0906.1346.

[25] B. Sotomayor et al. 2009. Virtual infrastructure management in private and hybrid clouds, IEEE Internet Computing, pp.14-22, 2009.

[26] B. Hindman et al. 2010. Mesos: A platform for fine-grained resource sharing in the data center, in Proc. NSDI 2011.

[27] M. Zaharia et al. 2010. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. in Proc. of EuroSys’10. ACM, New York, NY, 265-278.

[28] M. Isard et al. 2009. Quincy: fair scheduling for distributed computing clusters. in Proc. of SOSP ’09. ACM, New York, NY, 261-276.

[29] J. Hamilton. 2009. Internet-scale service infrastructure efficiency. SIGARCH Comput. Archit. News 37, 3 (Jun. 2009), 232-232.

[30] D. Thain et al. Distributed Computing in Practice: The Condor Experience. Concurrency and Computation: Practice and Experience,17(2):323-356, February 2005.

[31] U. Hoelzle et al. 2009. The Datacenter as a Computer: an Introduction to the Design of Warehouse-Scale Machines. 1st. Morgan and Claypool Publishers.

[32] P. Wang et al. 2010. Transformer: A New Paradigm for Building Data-Parallel Programming Models. IEEE Micro 30, 4 (Jul. 2010), 55-64.

[33] L. Rodero-Merino et al. 2010. From infrastructure delivery to service management in clouds. Future Gener. Comput. Syst. 26, 8 (Oct. 2010), 1226-1240.

[34] J. Zhan et al. 2010. PhoneixCloud: Provisioning Runtime Environments for Heterogeneous Cloud Workloads. Technical Report. http://arxiv.org/abs/1003.0958v1.

[35] S. Govindan, et al. 2009. Statistical profiling-based techniques for effective power provisioning in data centers. in Proc. of EuroSys'09. ACM, New York, NY, 317-330.

[36] P. Padala et al. Automated control of multiple virtualized resources. in Proc. EuroSys'09.

[37] B. Urgaonkar et al. 2002. Resource overbooking and application profiling in shared hosting platforms, in Proc. OSDI'02, Boston, MA.

[38] Z. Jia et al. BigDataBench: a Benchmark Suite for Big Data Applications in Data Centers. ICT Technical Report.

[39] J. Zhan et al. High Volume Throughput Computing: Identifying and Characterizing Throughput Oriented Workloads in Data Centers. in Proc. LSPP 2012 in conjunction with IPDSP 2012.

[40] A. Verma et al. 2008. Power-aware dynamic placement of HPC applications. in Proc. ICS '08.

[41] J. Chase et al. Managing energy and server resources in hosting centers. in Proc. of SOSP'01.

[42] C. D Patel et al. Cost Model for Planning, Development and Operation of a Data Center, Hewlett-Packard Laboratories Technical Report, PL-2005-107(R.1), June 9, 2005

[43] J. Moreira et al. The Case for Full-Throttle Computing: An Alternative Datacenter Design Strategy, IEEE Micro, vol. 30, no. 4, pp.25-28, July/Aug. 2010

[44] J. Karidis et al. 2009. True value: assessing and optimizing the cost of computing at the data center level. in Proc. of CF '09. ACM, New York, NY, 185-192.

[45] W. Iqbal et al. 2010. SLA-Driven Dynamic Resource Management for Multi-tier Web Applications in a Cloud. in Proc. of CCGrid' 10, pp.832-837.

[46] P. Campegiani et al. 2009. A General Model for Virtual Machines Resources Allocation in Multi-tier Distributed Systems. in Proc. of ICAS' 09.

[47] I. Goiri et al. 2010. Characterizing Cloud Federation for Enhancing Providers' Profit. in Proc. of Cloud' 10, pp.123-130

[48] W. Vogels, Beyond Server Consolidation, ACM Queue, vol. 6, no.1, 2008, pp. 20-26.

[49] X. Fan et al. 2007. Power provisioning for a warehouse-size computer. in Proc. of ISCA' 07.

[50] L. Rodero-Merino et al. 2010. From infrastructure delivery to service management in clouds. Future Gener. Comput. Syst. 26, 8 (Oct. 2010), 1226-1240.

[51] J. Choi et al. 2010. Power Consumption Prediction and Power- Aware Packing in Consolidated Environments. Computers, IEEE Transactions on , vol.59, no.12, pp.1640-1654, Dec. 2010

[52] A. Benoit et al. Scheduling Concurrent Bag-of-Tasks Applications on Heterogeneous Platforms. Computers, IEEE Transactions on, vol.59, no.2, pp.202-217, Feb. 2010

[53] P. Mart´ı et al. 2009. Draco: Efficient Resource Management for Resource-Constrained Control Tasks. IEEE Trans. Comput. 58, 1 (January 2009), 90-105.

[54] H. Xi et al. Characterization of Real Workloads of Web Search Engines. in Proc. IISWC 2011.

[55] G. S. Sadhasivam et al. Design and Implementation of a Two Level Scheduler for HADOOP Data Grids. Int. J. of Advanced Networking and Applications 295 Volume: 01, Issue: 05, Pages:295-300 (2010)

[56] http://www.sics.se/ aeg/report/node6.html

[57] C. A. Waldspurger et al. Lottery scheduling: Flexible proportional-share resource management. in Proc. OSDI 1994.

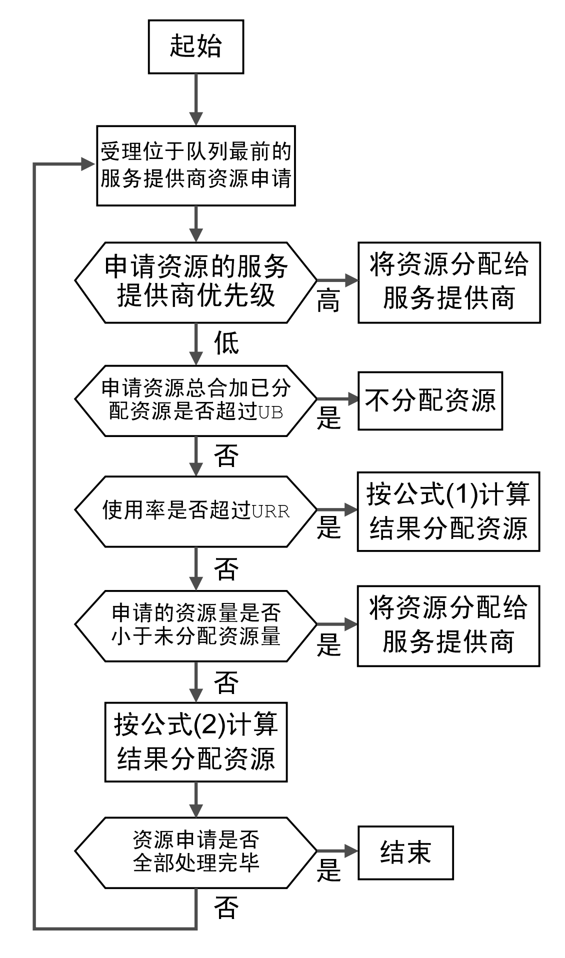
[58] W. Jin et al. Interposed proportional sharing for a storage service utility. in Proc. SIGMETRICS 2004.

[59] C. Luo et al. CloudRank-D: benchmarking and ranking cloud computing systems for data processing

applications. Frontiers of Computer Science (2012) 6: 347-362 , August 01, 2012

[60] C. Yanpei et al. The Case for Evaluating MapReduce Performance Using Workload Suites. International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) 2011.

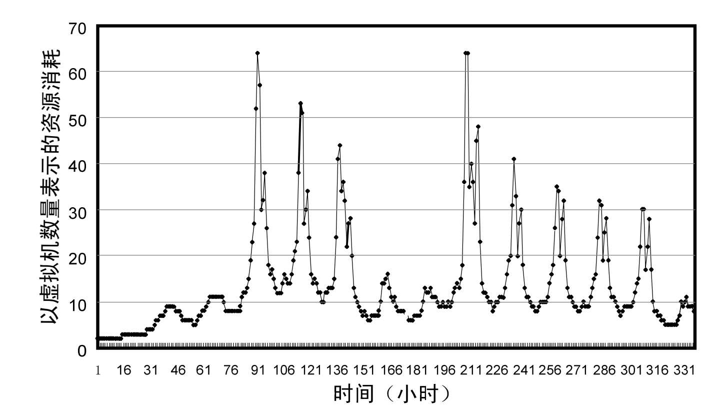
附录A 系统资源供应的运行时算法



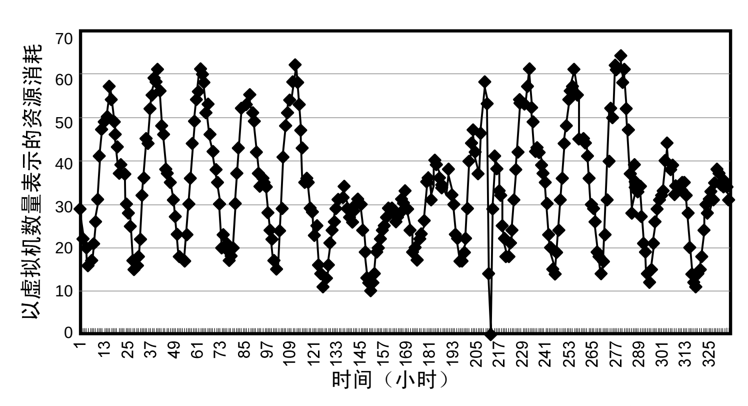
附录B 性能评价

B. 1 三个资源消耗日志

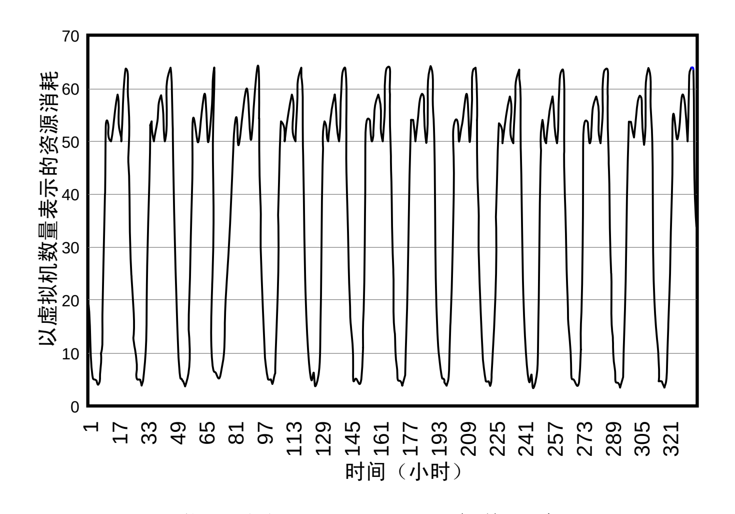
图 1、图 2 和图 3 分别显示 World Cup、ClarkNet、SEARCH 负载资源消耗数据。所有三个资源消耗的日志中平均资源需求和尖峰资源需求（峰值的资源的需求是 64 个虚拟机）的比率都很高。其中 SEARCH 日志由重复一个匿名 24 小时日志[54]放大 14 倍构造成的，以保持它和 World Cup 和 ClarkNet 的日志等长。



附录B图1.　世界杯两周期间的资源跟踪日志



附录B图2.　两周的ClarkNet 资源跟踪日志



附录B图3.　SEARCH负载日志

附录B表1.　6个真实负载对应不同资源提供商的系统指标

负载	系统	峰值资源消耗（节点数）	有效资源消耗（节点数×小时）	资源消耗总量（节点数×小时）
6个负载	非协作	6126	677190	687441
6个负载	PhoenixCloud	5019	563975	571157

附录B表2.　负载NASA iPSC对应不同服务提供商的系统指标

负载	系统	完成作业数	平均周转时间（秒）	资源消耗（节点数×小时）
6个负载	非协作	2603	573	54118
6个负载	PhoenixCloud	2603	741	36258

附录B表3.　负载SDSC BLUE对应不同服务提供商的系统指标

负载	系统	完成作业数	平均周转时间(秒)	资源消耗（节点数×小时）
6个负载	非协作	2657	1975	35838
6个负载	PhoenixCloud	2654	2607	32091

附录B表4.　LLNL Thunder负载对应不同服务提供商的系统指标

负载	系统	完成作业数	平均周转时间(秒)	资源消耗（节点数×小时）
6个负载	非协作	7273	2465	339416
6个负载	PhoenixCloud	7272	2844	288323

B.2 无限资源条件下的基本模拟实验

我们使用在 5.1 节介绍的主文件执行了负载的追踪实验。在这个实验中，我们假定资源提供商所拥有的资源足以满足所有的服务提供商，这意味着由服务提供商发出的每个资源请求相对于资源供应商所拥有的资源总量是非常小的。5.1 节介绍的主文件中的六个负载追踪共形成三种组合。我们用 Comb_ 1 代表（IPSC，WorldCup），Comb_2 代表（SDSC，Clark），Comb_3 代表（LLNL，SEARCH）。对于 Comb_1，（$\boldsymbol{PRC_A}$**,** $\boldsymbol{PRC_B}$）是（128，128）；对于 Comb_2，（$\boldsymbol{PRC_A}$**,** $\boldsymbol{PRC_B}$）是（144 ，128）;对于 Comb_3，（$\boldsymbol{PRC_A}$**,** $\boldsymbol{PRC_B}$）是（1002，896)。附录 B.3 中，我们探讨了用（$\boldsymbol{PRC_A}$**,** $\boldsymbol{PRC_B}$）表示的不同规模对性能指标的影响。我们设置 PhoenixCloud 中的基线参数如下：[R0.5 / E100000 / U1.2 / V0.1 / L60]。 E100000 意味着，所有的服务提供商有更高的资源上限。由于资源提供商有无限的资源，我们分别设置资源预订阈值 RBT 和使用率参考值 URR 为 1。(参见第三章主要文件中解释)。

每个实验进行 6 次，我们的报告取 6 次实验的平均值。表 1、表 2、表 3 和表 4 总结了关于资源提供商和服务提供者的实验结果。在类 EC2+RightScale 系统中，资源提供商为 Web 服务和并行批处理作业负载独立供应资源。同时，在 PhoenixCloud 系统中，Web 服务负载比并行批处理作业有更高的优先级，因此 Web 服务的资源请求将被立即满足。所以，在这两个系统中对于两个 Web 服务负载的追踪有着相同或相近的性能指标。因此，对于服务提供商和最终用户，我们只观测了并行批处理作业负载的性能指标。

对于资源提供商，资源消耗总量是有效资源消耗和管理开销的总和。我们在测试平台上部署了真实的 PhoenixCloud 系统，分配和回收一个节点的平均开销是 225 秒，步骤包括部署操作系统，停止和卸载以前的运行时环境包，安装和启动新的运行时环境包。我们根据下面公式计算管理开销：

管理开销=（累计的调整资源的计数）×（平均安装时间）

从表 1、表 2、表 3 和表 4 中，我们可以得出以下几点：

对于类 EC2 + RightScale 系统，PhoenixCloud 系统中所允许的最低系统容量是由尖峰资源消耗决定的，六个异构负载的尖峰资源消耗降低了 18%，每个服务提供商运行并行批处理作业的吞吐量接近于使用类 EC2 +RightScale 系统，并且每个服务提供商运行并行批处理作业负载都节省了资源消耗；对于每个终端用户，平均周转时间有少量增加，分别为 29%，24%和 15%。

得出这一实验结果有三方面原因：（1）.在 PhoenixCloud 系统中，因为并行批处理作业可以排队，在资源上限以外的资源将会被用于满足需要立即做出响应的 Web 服务的资源请求，但在类 EC2 系统中，由于充分利用了统计复用技术，每个批处理作业所需的资源将会被立即供应。因此，使用 PhoenixCloud 系统，资源提供商可以比使用类 EC2 + RightScale 系统减少尖峰资源消耗。（2）.在 PhoenixCloud 中，以非独占式的模式运行异构负载的每组两个服务提供商共享使用上下限之间的资源，通过统计复用技术，不仅可以防止他们资源浪费，同时也降低了尖峰资源消耗。（3）.对于资源下界内的资源，只有两个协同的服务提供商共享使用，而不需要动态的要求或释放，因此，我们的方法降低了管理开销。

B.3 不同负载规模的实验

本节讨论不同负载规模 ***(PRC_A, PRC_B)*** 对资源供应者、服务提供商和最终用户的影响。负载组合是（（iPSC, Clark），（SDSC, WorldCup））。Phoenix Cloud 系统的基线配置为[R0.5\E400\U1.2\V0.1\L60]。我们假定系统容量是无限的，并且资源预订阈值 RBT 和使用率参考值 URR 分别设置为 1。在下面的实验，并行的批处理作业负载的规模与附录 B.2 节相同，而我们用不同的常量扩展了 Web 服务负载的追踪。

附录B表5. 不同负载规模下的资源提供商指标

负载 ((PRCA; PRCB), (PRCA; PRCB))	节约的峰值资源损耗	节约的资源损耗总量
((128,128),(144,128))	66.3%	15.1%
((128,64),(144,68))	62.5%	22.1%
((128,256),(144,256))	62.5%	11.9%

附录B表6. NASA和Clark在不同负载规模下的并行批处理作业

负载 (*PRCA; RCB*)	完成作业数	平均周转时间	资源消耗
(128,64)	2603	780	36808
(128,128)	2603	979	36518
(128,256)	2603	2686	36287

附录B表7. SDSC和World在不同负载规模下的并行批处理作业

负载 (*PRCA; RCB*)	完成作业数	平均周转时间	资源消耗
(144,64)	2656	2624	33716
(144,128)	2656	2559	35093
(144,256)	2652	2443	39600

表 5、表 6、表 7 显示了实验结果。我们可以看到：第一，对于资源提供商，相比类 **EC2+RightScale** 系统，尖峰资源消耗降低了 60%左右，此结果与 ***PRC_B*** 和 ***PRC_A*** 的比值无关；当 ***PRC_B*** 和 ***PRC_A*** 的比值增加时，资源消耗总量降低。其次，当 Web 服务负载有很大的资源需求时，***PRC_B*** 和 ***PRC_A*** 比值的增加，将导致并行批处理作业很大的延迟，这是因为 Web 服务的优先级高于并行批处理作业。如图 1 和图 2 所示， 与 WorldCup 相比，ClarkNet 变化比较平缓，但后者有更持久的资源需求，因此当 ***PRC_B*** 与 ***PRC_A*** 的比值增加时，ClarkNet 负载导致并行批处理作业平均周转时间拉长。

B.4 不同参数的影响

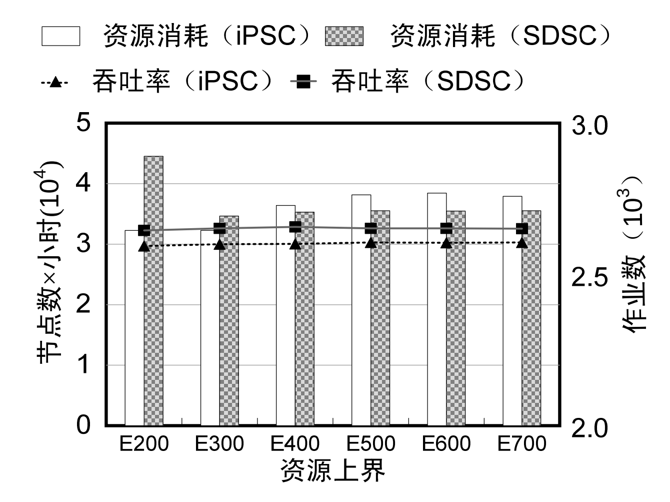
本节探讨 PhoenixCloud 系统中不同的效果参数。由于篇幅的限制，我们无法一一呈现

数据的所有参数组合情况，我们采用将一个或两个参数变化而其他参数保持相同基线配置[R0.5/E400/U1.2/V0.1/L60]。我们假定资源容量是无限的，分别设置资源预订阈值 RBT 和使用率参考值 URR 为 1。

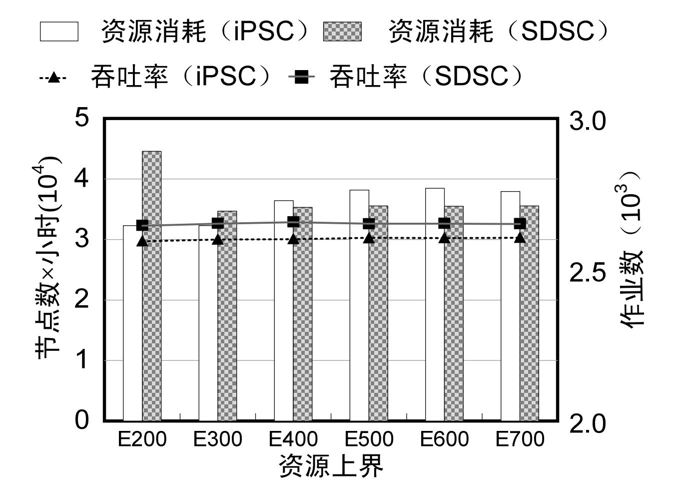
B.4.1 资源上界的影响

图 4 和图 5 显示了对于资源提供商和服务提供商，资源上界调整带来的影响。

我们可以看到，当资源上界与 PRC_B 和 PRC_A 的总和的比值小于 2 时，尖峰资源消耗增加。当资源上界与 PRC_B 和 PRC_A 的总和的比值接近 1 时，PhoenixCloud 系统将会有更高的总资源消耗和更低的吞吐量。在实验中，我们建议设置资源上界与 PRC_B 和 PRC_A 的总和的比值为 1.5。



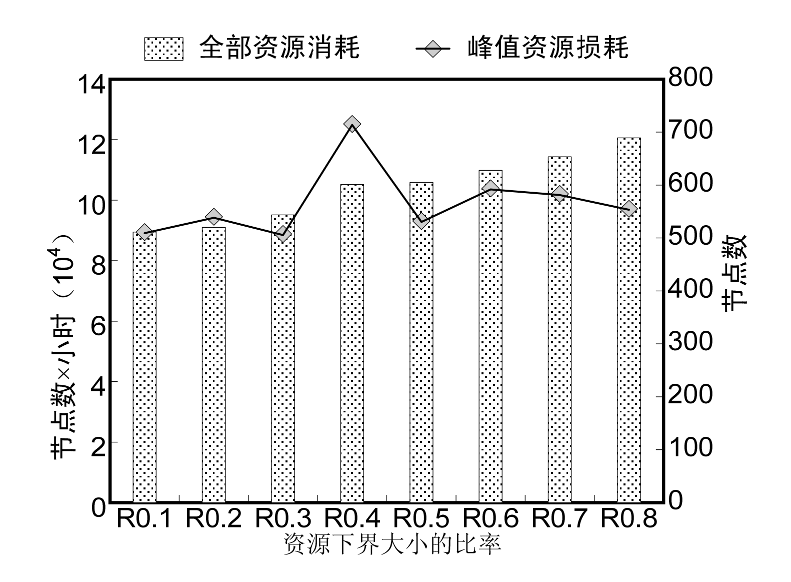
附录B图4. 不同资源上界的资源提供商指标



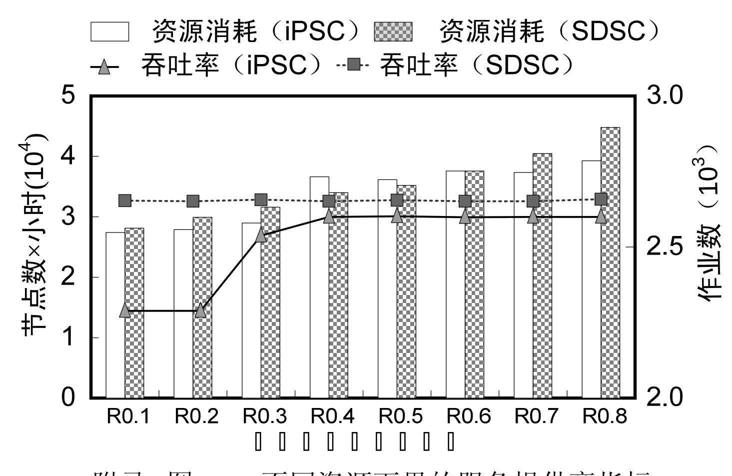
附录B图5. 不同资源上界的服务提供商指标

B.4.2 资源下界的影响

图 6 和图 7 显示了资源下界调整对资源提供商和服务提供商的影响。我们可以看到，资源提供商的总资源消耗与资源下界成正比，而资源下界对尖峰资源消耗的影响不大（R0.4 除外）；对于服务提供商，资源消耗与资源下界成正比，在同一时间，较低的资源下界会导致一些负载吞吐量下降（对于 iPSC 负载，小于 0.4）。在实验中，我们建议将 R 的值设置为 0.5。



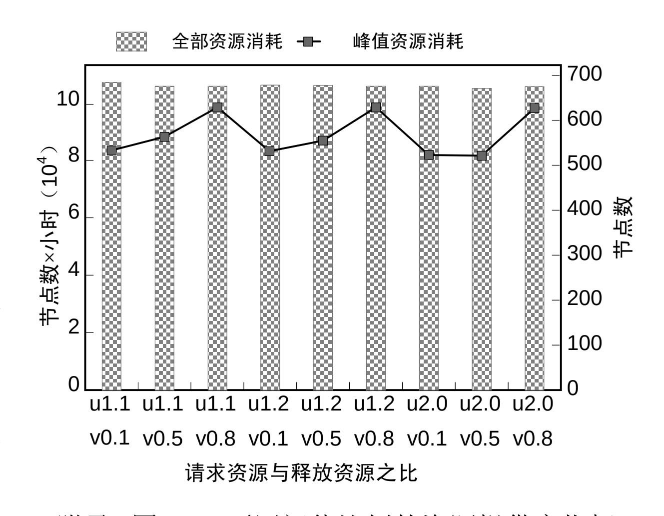
附录B图6.　不同资源下界的资源提供商指标



附录B图7.　不同资源下界的服务提供商指标

B.4.3 请求资源与释放资源比率阈值的影响

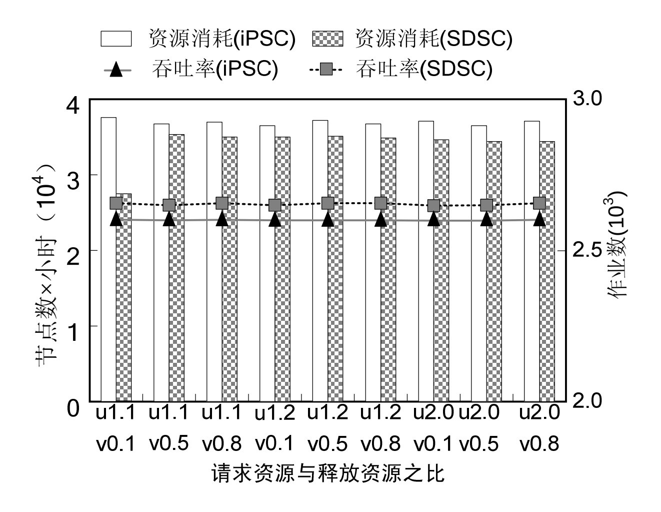
图 8 和图 9 显示了请求/释放资源的阈值比率对于资源提供商和服务供应者的影响。我们可以看到：请求和释放资源的比率阈值对资源消耗总量和服务提供商的影响不大。在同一时间，尖峰资源消耗随着释放资源的阈值比例的增加而增长。在实验中，我们建议分别设置 U 和 V 为 1.2 和 0.1。



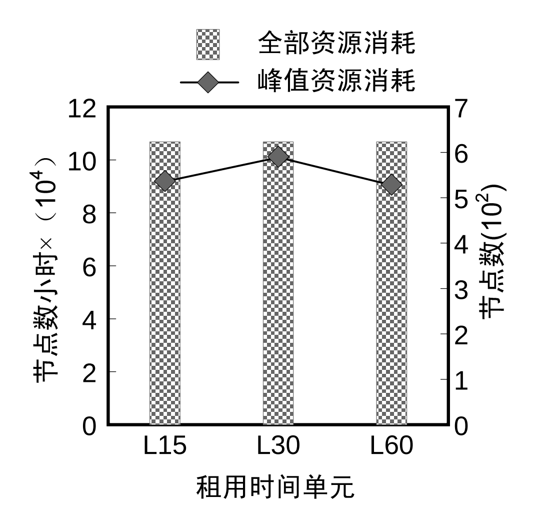
附录B图8.　不同阈值比例的资源提供商指标

租赁资源的时间单元的影响：我们把租赁时间单元从 15 分钟改到 60 分钟。图 10 中的 L15、L30 和 L60 分别意谓着租赁时间单元为 15、30 和 60 分钟。由我们的实验可以得出结论：租

赁资源的时间单元对资源供应者影响不大。



附录B图9.　比例阈值服务提供商指标



附录B图10.　在不同时间单元释放资源，资源提供商指标

B.5 合成 MapReduce 负载

附录B表8.　Facebook和我们的负载中，按照map任务数量来算，作业大小的影响

分组	在 facebook 的 Map 数	在 facebook 的作业	在负载中的 Map 数	在负载中的作业数
1	1	39%	1	201
2	2	16%	2	81
3	3~20	14%	10	65
4	21~60	9%	50	32
5	61~150	6%	100	27
6	151~300	6%	200	28
7	301~500	4%	400	19
8	>500	7%	1300	24

根据对Facebook2009年10月为期一周的日志记录[61]的分析，我们通过选择不同类型和大小的MapReduce作业合成了MapReduce负载。合成的MapReduce的负载包括500个作业。首先，

根据对Facebook日志的分析，合成的负载的任务数量从1到1300不等。表8显示了Facebook的日志和我们的合成负载。第二步，如表9所示，我们合成的MapReduce负载的作业类型是从不同类别Hadoop应用选择的，例如，分析和统计类别的词频计算（Wordcount），排序算法类别的排序和TeraSort，计算类应用BBP等等。第三步，我们通过产生从1到500的随机数序列，生成作业提交顺序。第四，根据对Facebook日志的分析，我们设定作业提交后的到达时间间隔遵从均值为140秒的指数分布。

附录B表9. 在合成MapReduce负载中，作业类型和大小的影响

分组	作业类型	每类的作业数	Map任务数	作业运行次数
1	检索	83	10	26
			50	15
			100	9
			200	19
			1300	14
2	BBP计算圆周率	66	10	39
			50	17
			1300	10
3	Terasort*	218	1	118
			2	81
			100	10
			200	9
4	WordCount※	110	1	83
			100	8
			400	19

* Hadoop的基准测试程序
※词频度计算

作者简介：
詹剑锋： 中科院计算所副研究员,主要研究数据中心计算和云计算. jfzhan@ncic.ac.cn
王 磊： 中科院计算所高级工程师,主要研究方向负载特征和应用驱动的系统设计
李晓娜： 中科院计算所客座硕士研究生,现在百度从事项目管理工作.
施巍松： 美国韦恩州立大学副教授.
翁楚良： 上海交通大学副教授
张文耀： 北京理工大学博士,讲师.
臧秀涛： 中科院计算所工程师
周俊平： 中科院计算所硕士研究生,现在工商银行上海开发中心工作.